Multi-timed Bisimulation for Distributed Timed Automata

Ortiz Vega, James Jerson; Schobbens, Pierre; Amrani, Moussa

Published in:
NASA Formal Methods - 9th International Symposium, NFM 2017 Moffett Field, Proceedings

DOI:
10.1007/978-3-319-57288-8_4

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Multi-Timed Bisimulation for Distributed Timed Automata

James Ortiz, Moussa Amrani, and Pierre-Yves Schobbens

Computer Science Faculty, University of Namur
{james.ortizvega, moussa.amrani, pierre-yves.schobbens}@unamur.be

Abstract. Timed bisimulation is an important technique which can be used for reasoning about behavioral equivalence between different components of a complex real-time system. The verification of timed bisimulation is a difficult and challenging problem because the state explosion caused by both functional and timing constraints must be taken into account. Timed bisimulation was shown decidable for Timed Automata (TA). Distributed TA and TA with Independent Clocks (icTA) were introduced to model Distributed Real-time Systems. They are a variant of TA with local clocks that may not run at the same rate. In this paper, we first propose to extend the theory of Timed Labeled Transition Systems to Multi-Timed Labeled Transition Systems, and relate them by an extension of timed bisimulation to multi-timed bisimulation. We prove the decidability of multi-timed bisimulation and present an EXPTIME algorithm for deciding whether two icTA are multi-timed bisimilar. For multi-timed bisimilarity, an extension of the standard refinement algorithm is described.

1 Introduction

Distributed Real-Time Systems (DTS) are increasing with the scientific and technological advances of computer networks. The high demand for computer networks has caused the development of new complex applications which benefit from the high performance and resources offered by modern telecommunications networks. Current researches in the area of DTS have emerged from the need to specify and analyze the behavior of these systems, where both distributed behavior and timing constraints are present. Formal verification methods, such as model checking, have been used to verify the correctness of complex DTS. Model checking over DTS becomes rapidly intractable because the state space often grows exponentially with the number of components considered. A technique to reduce the state space is to merge states with the same behaviour. For untimed systems, the notion of *bisimulation* [13] is classically used to this end, and its natural extension for real-time systems, *timed bisimulation*, was already shown decidable for Timed Automata (TA) [2, 12]. A timed automaton is a finite automaton augmented with real-valued clocks, represented as variables that increase at the same rate as time progresses. TA assume perfect clocks: all clocks have infinite precision and are perfectly synchronized. In this paper, we study

two variants of TA called Distributed Timed Automata (DTA) and Timed Automata with Independent Clocks (icTA) proposed by [11, 1, 16] to model DTS, where the clocks are not necessarily synchronized. TA have been used to model DTS such as Controller Area Network [14] and WirelessHART Networks [10]. But, TA, icTA and timed bisimulation are based on a sequential semantics of a Timed Labelled Transition Systems (TLTS), i.e., a run of a TLTS is given by a sequence of actions and timestamps.

Unfortunately, a sequential semantics does not describe completely the behavior of the DTS, because interactions between processes with their associated local clocks that are running at the same rate and distribution of the actions over the components are not considered. Also, model-checking and bisimulation equivalence algorithms have been implemented in tools [20][19] for the sequential semantics used by the model (e.g., TA, TLTS, etc). In contrast, behavioral equivalences for DTS have only been introduced in [3]. It is, however, not clear whether such equivalences agree with the distributed timed properties in DTS. Therefore, we propose an alternative semantics to the classical sequential semantics for TLTS and icTA: specifically, a run of a system in our alternative semantics is given by the sequences of pairs (action, tuples of timestamps). We propose an alternative semantics in order to be able to consider a semantics which expresses the distribution of the actions and timestamps over the components. With this alternative, it becomes possible to analyze the local behavior of the components *independently*, thus enhancing the expressiveness of the TLTS (and icTA). We introduce Multi-Timed Labelled Transition Systems (MLTS), an extension of classical TLTS in order to cope with the notion of multiple local times, and we propose efficient algorithms using refinement techniques [17].

Contributions. One of our main contributions is to incorporate a alternative semantics over sequential semantics for TLTS and icTA. Also, we extend the classical theory of timed bisimulation with the notion multi-timed bisimulation and their corresponding decision algorithms. We also present two algorithms:(i) a forward reachability algorithm for the parallel composition of two icTA, which will help us to minimize the state space exploration by our second algorithm, and (ii) a decision algorithms for multi-timed bisimulation using the zone-based technique [5]. Multi-timed bisimulation is a relation over local clocks (and processes), and cannot be computed with the standard partition refinement algorithm [17]. Instead, our algorithm successively refines a set of zones such that ultimately each zone contains only multi-timed bisimilar pairs of states. Furthermore, we show that our algorithm is EXPTIME-complete. Since TA are a special variant of icTA, our work conservatively extends the expressiveness of TA and TLTS; and since timed bisimulation over TA [20, 19] can be regarded as a special case of multi-timed bisimulation, our decision algorithms could potentially be used to analyze complex DTS.

Structure of the paper. After recalling preliminary notions in Section 2, we introduce our alternative semantics for icTA in Section 3, based on multi-timed words consumed by MLTS. Section 4 deals with bisimulation: we first define multi-timed bisimulation, by adapting the classical definition to MLTS, then

show its decidability by exhibiting an EXPTIME algorithm. Finally, Section 5 compares our work with existing contributions, and Section 6 concludes. Due to space constraints, some proofs are not given here, but stay available in a Technical Report available online [15].

2 Preliminaries

We describe in this section the notations needed for formally defining Timed Labelled Transition Systems (TLTS) and Timed Automata TA.

Timed Words. The set of all *finite words* over a finite alphabet of actions Σ is denoted by Σ^*. Let $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ respectively denote the sets of natural, real and nonnegative real numbers. A timed word [2] over Σ is a finite sequence $\theta = ((\sigma_1, t_1), (\sigma_2, t_2) \dots (\sigma_n, t_n))$ of actions paired with nonnegative real numbers (i.e., $(\sigma_i, t_i) \in \Sigma \times \mathbb{R}_{\geq 0}$) such that the timestamped sequence $t = t_1 \cdot t_2 \cdots t_n$ is nondecreasing (i.e., $t_i \leq t_{i+1}$). We sometimes define θ as the pair $\theta = (\sigma, t)$ with $\sigma \in \Sigma^*$ and t a sequence of timestamps with the same length.

Clocks. A clock is a real positive variable that increases with time. Let X be a finite set of clock names. A clock constraint $\phi \in \Phi(X)$ is a conjunction of comparisons of a clock with a natural constant c: with $x \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, >, \leq, \geq, =\}$, ϕ is defined by

$$\phi ::= true \mid x \sim c \mid \phi_1 \ \wedge \ \phi_2$$

A clock valuation $\nu \in \mathbb{R}^X_{\geq 0}$ over X is a mapping $\nu : X \to \mathbb{R}_{\geq 0}$. For a time value $t \in \mathbb{R}_{\geq 0}$, we note $\nu + t$ the valuation defined by $(\nu + t)(x) = \nu(x) + t$. Given a clock subset $Y \subseteq X$, we note $\nu[Y \to 0]$ the valuation defined as follows: $\nu[Y \leftarrow 0](x) = 0$ if $x \in Y$ and $\nu[Y \leftarrow 0](x) = \nu(x)$ otherwise. The projection of ν on Y, written $\nu\rfloor_Y$, is the valuation over Y containing only the values in ν of clocks in Y.

Timed Automata (TA). A TA is a tuple $\mathcal{B} = (\Sigma, X, S, s_0, \to_{ta}, I, F)$ where Σ is a finite alphabet, X a clock set, S a set of locations with $s_0 \in S$ the initial location and $F \subseteq S$ the set of (sink) final states, $\to_{ta} \subseteq S \times \Sigma \times \Phi(X) \times 2^X \times S$ is the automaton's transition relation, $I : S \to \Phi(X)$ associates to each location a clock constraint as invariant. For a transition $(s, \phi, a, Y, s') \in \to_{ta}$, we classically write $s \xrightarrow{\phi, a, Y} s'$ and call s and s' the source and target location, ϕ is the guard, a the action or label, Y the set of clocks to be reset. During the execution of a TA $\mathcal{B}$, a *state* is a pair $(s, \nu) \in S \times \mathbb{R}^X_{\geq 0}$, where s denotes the current state with its accompanying clock valuation ν, starting at s_0, ν_0 where ν_0 maps each clock to 0. We only consider *legal* states, i.e. states that satisfy $\nu \vDash I(s)$ (i.e. valuations that map clocks to values that satisfy the current state's invariant).

Timed Transition System (TLTS). The transition system $\mathsf{TLTS}(\mathcal{B})$ generated by $\mathcal{B}$ is defined by $\mathsf{TLTS}(\mathcal{B}) = (Q, q_0, \Sigma, \to_{tlts})$, where Q is a set of legal states over $\mathcal{B}$ with initial state $q_0 = (s_0, \nu_0)$, Σ a finite alphabet and $\to_{tlts} \subseteq Q \times (\Sigma \uplus \mathbb{R}_{\geq 0}) \times Q$ is the TLTS transition relation defined by : (a) Delay transition:

$(s,\nu) \xrightarrow{t} (s,\nu+t)$ for some $t \in \mathbb{R}_{\geq 0}$, iff $\nu + t \vDash I(s)$, (b) Discrete transition: $(s,\nu) \xrightarrow{a} (s',\nu')$, iff $s \xrightarrow{\phi,a,Y} s'$, $\nu \vDash \phi$, $\nu' \vDash \nu[Y \to 0]$ and $\nu' \vDash I(s')$.

3 An Alternative Semantics for DTA

In this section, we define an alternative semantics (which we will call multi-timed semantics) for icTA as opposed to the mono-timed semantics of [1]. The main problem with the semantics of [1] is that they use the reference time. The benefits of this new definition are threefold. First, the multi-timed semantics preserves the untimed language of the icTA. Second, the multi-timed semantics can work with multi-timed words. Third, the region equivalence defined in [1] could form a finite time-abstract bisimulation on the multi-timed semantics. Hence, the multi-timed semantics allows to build a region automaton that accepts exactly Untime($\mathcal{L}(\mathcal{A})$) for all icTA $\mathcal{A}$ [1]. Thus, we extend TLTS and icTA to their multi-timed version.

3.1 Multi-Timed Actions

Let $Proc$ be a non-empty set of processes, then, we denote by $\mathbb{R}_{\geq 0}^{Proc}$ the set of functions from $Proc$ to $\mathbb{R}$, that we call *tuples*. A tuple $\boldsymbol{d} \in \mathbb{R}_{\geq 0}^{Proc}$ is smaller that $\boldsymbol{d}'$, noted, $\boldsymbol{d} < \boldsymbol{d}'$ iff $\forall i \in Proc\ \boldsymbol{d}_i \leq \boldsymbol{d}'_i$ and $\exists i \in Proc\ \boldsymbol{d}_i < \boldsymbol{d}'_i$. A Monotone Sequence of Tuples (MST) is a sequence $\boldsymbol{d} = \boldsymbol{d}_1\boldsymbol{d}_2 \cdots \boldsymbol{d}_n$ of tuples of $\mathbb{R}_{\geq 0}^{Proc}$ where : $\forall j \in 1 \cdots n-1$, $\boldsymbol{d}_j \leq \boldsymbol{d}_{j+1}$. A multi-timed word on Σ is a pair $\theta = (\sigma, \boldsymbol{d})$ where $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$ is a finite word $\sigma \in \Sigma^*$, and $\boldsymbol{d} = \boldsymbol{d}_1\boldsymbol{d}_2 \ldots \boldsymbol{d}_n$ is a MST of the same length. This is the analog of a *timed word* (or *multi-timed action*) [2]. A *multi-timed word* can equivalently be seen as a sequence of pairs in $\Sigma \times \mathbb{R}_{\geq 0}^{Proc}$.

3.2 Multi-Timed Labeled Transition Systems

Our multi-timed semantics is defined in terms of *runs* that record the state and clock values at each transition points traversed during the consumption of a *multi-timed* word. Instead of observing actions at a global time, a multi-timed word allows to synchronise processes on a common action that may occur at a specific process time.

Definition 1 (Multi-Timed Labelled Transition System). *A Multi-Timed Labelled Transition System (MLTS) over a set of processes Proc is a tuple* $\mathcal{M} = (Q, q_0, \Sigma, \to_{mlts})$ *such that:* (i) Q *is a set of states.* (ii) $q_0 \in Q$ *is the initial state.* (iii) Σ *is a finite alphabet.* (v) $\to_{mlts} \subseteq Q \times (\Sigma \uplus \mathbb{R}_{\geq 0}^{Proc}) \times Q$ *is a set of transitions.*

The transitions from state to state of a MLTS are noted in the following way: (i) A transition (q,a,q') is denoted $q \xrightarrow{a} q'$ and is called a *discrete transition*, if $a \in \Sigma$ and $(q,a,q') \in \to_{mlts}$, (ii) A transition $(q,\boldsymbol{d},q')$ is denoted $q \xrightarrow{\boldsymbol{d}} q'$ and is called a *delay transition*, if $\boldsymbol{d} \in \mathbb{R}_{\geq 0}^{Proc}$ and $(q,\boldsymbol{d},q') \in \to_{mlts}$.

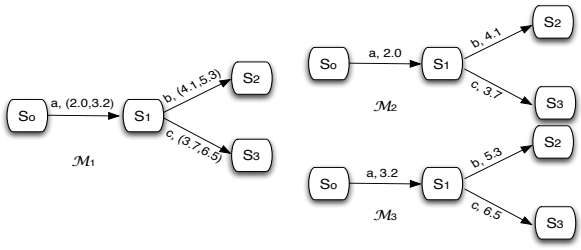

Fig. 1. Multi-Timed and Timed Labelled Transition Systems

A run of $\mathcal{M}$ can be defined as a finite sequence of moves, where discrete and continuous transitions alternate: $\rho = q_0 \xrightarrow{\boldsymbol{d}_1} q'_0 \xrightarrow{a_1} q_1 \xrightarrow{\boldsymbol{d}_2} q'_1 \xrightarrow{a_2} q_2 \dots q_{n-1} \xrightarrow{\boldsymbol{d}_{n-1}} q'_{n-1} \xrightarrow{a_{n-1}} q_n$, where $\forall\ 0 \leq i \leq n-1, q_i \in Q, \forall j \leq n-1, \boldsymbol{d}_j \in \mathbb{R}^{Proc}_{\geq 0}, q'_j \in Q$ and $a_j \in \Sigma$. The *multi-timed word* of ρ is $\theta = ((a_1, \boldsymbol{t}_1), (a_2, \boldsymbol{t}_2) \dots, (a_n, \boldsymbol{t}_n))$, where $\boldsymbol{t}_i = \sum_{j=1}^{i} \boldsymbol{d}_j$. A multi-timed word θ is *accepted* by $\mathcal{M}$ iff there is a maximal initial run whose multi-timed word is θ. The *language* of $\mathcal{M}$, denoted $\mathcal{L}(\mathcal{M})$, is defined as the set of multi-timed words accepted by some run of $\mathcal{M}$. Note that MLTS are a proper generalisation of TLTS: each TLTS can be seen as a MLTS with a single process and conversely.

For example, consider the two transition systems in Figure 1: a MLTS on the left ($\mathcal{M}_1$) and two TLTS on the right ($\mathcal{M}_2$ and $\mathcal{M}_3$) with the finite input alphabet $\Sigma = \{a, b, c\}$. In brief, $\mathcal{M}_2$ and $\mathcal{M}_3$ could be considered as the projection of $\mathcal{M}_1$ on the case of process 1 and 2.

3.3 A Multi-timed Semantics for icTA

DTA [11, 1] consist of a number of local timed automata. In [1], DTA are not much studied. Instead, their product is first computed, giving rise to the class of icTA ($\mathcal{A} = (\mathcal{B}, \pi)$, where $\mathcal{B}$ is a TA and π is a function maps each clock to a process).

Given $\pi : X \to Proc$, a clock valuation $\nu : X \to \mathbb{R}_{\geq 0}$ and $\boldsymbol{d} \in \mathbb{R}^{Proc}_{\geq 0}$: the valuation $\nu +_\pi \boldsymbol{d}$ is defined by $(\nu +_\pi \boldsymbol{d})(x) = \nu(x) + \boldsymbol{d}_{\pi(x)}$ for all $x \in X$. A Rate is a tuple $\tau = (\tau_q)_{q \in Proc}$ of local time functions. Each local time function τ_q maps the reference time to the time of process q, i.e, $\tau_q : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. The functions τ_q must be continuous, strictly increasing, divergent, and satisfy $\tau_q(0) = 0$. The set of all these tuples τ is denoted by *Rates*.

The operational semantics of an icTA has been associated to a sequential semantics. A run of an icTA $\mathcal{A}$ for $\tau \in Rates$ with a sequential semantics as a sequence $(s_1, \nu_1) \xrightarrow{t_1, a_1} (s_2, \nu_2) \xrightarrow{t_2, a_2} (s_2, \nu_3) \dots (s_{n-1}, \nu_{n-1}) \xrightarrow{t_{n-1}, a_{n-1}} (s_n, \nu_n)$ where $\forall 1 \leq i \leq n$, $s_i \in S$ and $\forall j \leq n-1$, $t_j \in \mathbb{R}_{\geq 0}$ and $a_j \in \Sigma$. Here, we want to associate operational semantics of a icTA to a MLTS.

Definition 2. *Let $\mathcal{A}$ be an icTA and $\tau \in Rates$. Our multi-timed semantics of the icTA $\mathcal{A}$ is given by a MLTS over Proc, denoted by MLTS($\mathcal{A}, \tau$) = ($Q, q_0, \Sigma, \rightarrow_{mlts}$).*

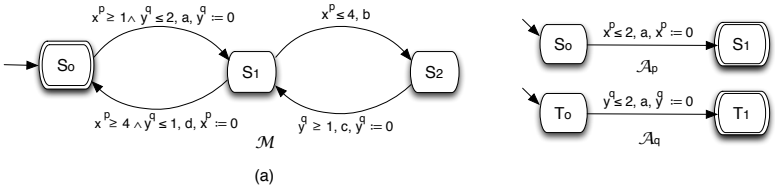

Fig. 2. (a) An icTA $\mathcal{M}$, (b) An counter example of Multi-timed Bisimulation

The set of states Q consists of triples composed of a location, a clock valuation and lastly the reference time: $Q = \{(s, \nu, t) \in S \times \mathbb{R}^X_{\geq 0} \times \mathbb{R}_{\geq 0} \mid \nu \models I(s)\}$. The starting state is $q_0 = (s_0, \nu_0, 0)$, where ν_0 is the valuation that assigns 0 to all the clocks. Σ is the alphabet of $\mathcal{A}$. The transition relation $\rightarrow_{mlts}$ is defined by:

(i) A transition $(q_i, \boldsymbol{d}, q_i')$ is denoted $q_i \xrightarrow{\boldsymbol{d}} q_i'$, and is called a *delay transition*, where $q_i = (s_i, \nu_i, t_i)$, $q_i' = (s_i, \nu_i +_\pi \boldsymbol{d}, t_{i+1})$, $\boldsymbol{d} = \tau(t_{i+1}) - \tau(t_i)$ and $\forall t \in [t_i, t_{i+1}] : \nu_i +_\pi (\tau(t) - \tau(t_i)) \models I(s_i)$.

(ii) A transition (q_i, a, q_{i+1}) is denoted $q_i \xrightarrow{a} q_{i+1}$, and is called a *discrete transition*, where $q_i = (s_i, \nu_i, t_i)$, $q_{i+1} = (s_{i+1}, \nu_{i+1}, t_{i+1})$, $a \in \Sigma$, there exists a transition $(s_i, a, \phi, Y, s_{i+1}) \in \rightarrow_{\mathsf{ic}}$, such that $\nu_i \models \phi$, $\nu_{i+1} = \nu_i[Y \rightarrow 0]$, $\nu_{i+1} \models I(s_{i+1})$, $t_i = t_{i+1}$.

In Definition 2, we have introduced a multi-timed semantics for icTA, following ideas of [1]. A run of an icTA $\mathcal{A}$ for $\tau \in Rates$ with our multi-timed semantics is an initial path in MLTS$(\mathcal{A}, \tau)$ where discrete and continuous transition alternate. A multi-timed word is accepted by $\mathcal{A}$ for $\tau \in Rates$ iff it is accepted by MLTS$(\mathcal{A}, \tau)$.

Example 1. The Figure 2(a) shows an icTA $\mathcal{M}$ with the finite input alphabet $\Sigma = \{a, b, c, d\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$ and $\tau = (2t, t)$ i.e. $\tau_p(t) = 2t$ and $\tau_q(t) = t$. A run of $\mathcal{M}$ on multi-timed word $\theta = ((a, (2.0, 1.0))(b, (3.0, 1.5))(c, (4.2, 2.1))(d, (6.0, 3.0)))$ is given by ρ $(S_0, [x^p = 0.0, y^q = 0.0], 0.0) \xrightarrow{(2.0,1.0)} (S_0, [x^p = 2.0, y^q = 1.0], 1.0) \xrightarrow{a} (S_1, [x^p = 2.0, y^q = 0.0], 1.0) \xrightarrow{(1.0,0.5)} (S_1, [x^p = 3.0, y^q = 1.5], 1.5) \xrightarrow{b} (S_2, [x^p = 3.0, y^q = 1.5], 1.5) \xrightarrow{(1.2,0.6)} (S_2, [x^p = 4.2, y^q = 1.1], 2.1) \xrightarrow{c} (S_1, [x^p = 4.2, y^q = 0.0], 2.1) \xrightarrow{(1.8,0.9)} (S_1, [x^p = 6.0, y^q = 0.9], 3.0) \xrightarrow{c} (S_0, [x^p = 0.0, y^q = 0.9], 3.0)$.

4 Multi-Timed Bisimulation

From a distributed approach, a DTS consist of several processes with their associated local clocks that are not running at the same rate. Thus, in order to formalize preservation of distributed timed behavior, we extend the classical definition of timed bisimulation [9] towards a multi-timed semantics. Our motivation for extending the classical definition of timed bisimulation is twofold: first, efficient algorithms checking for timed and time-abstract bisimulation have

been discovered [12][19]. Nonetheless, these algorithms are based on sequential semantics (i.e, TLTS and TA). Second, verifying the preservation of distributed timed behavior in DTS could be used to master the combinatorial explosion of the size of the model due to the composition of the processes.

4.1 Strong Multi-timed Bisimulation

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MLTS over the same set of actions Σ and processes $Proc$. Let $Q_{\mathcal{M}_1}$ (resp., $Q_{\mathcal{M}_2}$) be the set of states of $\mathcal{M}_1$ (resp., $\mathcal{M}_2$). Let $\mathcal{R}$ be a binary relation over $Q_{\mathcal{M}_1} \times Q_{\mathcal{M}_2}$. We say that $\mathcal{R}$ is a strong multi-timed bisimulation whenever the following transfer property holds (note that technically this is simply strong bisimulation over $\Sigma \uplus \mathbb{R}_{\geq 0}^{Proc}$):

Definition 3. *A strong multi-timed bisimulation over MLTS $\mathcal{M}_1$, $\mathcal{M}_2$ is a binary relation $\mathcal{R} \subseteq Q_{\mathcal{M}_1} \times Q_{\mathcal{M}_2}$ such that, for all $q_{\mathcal{M}_1} \mathcal{R} q_{\mathcal{M}_2}$, the following holds:*

(i) For every $a \in \Sigma$ and for every discrete transition $q_{\mathcal{M}_1} \xrightarrow{a}_{\mathcal{M}_1} q'_{\mathcal{M}_1}$, there exists a matching discrete transition $q_{\mathcal{M}_2} \xrightarrow{a}_{\mathcal{M}_2} q'_{\mathcal{M}_2}$ such that $q'_{\mathcal{M}_1} \mathcal{R} q'_{\mathcal{M}_2}$ and symmetrically.

(ii) For every $\boldsymbol{d} = (d_1, \ldots, d_n) \in \mathbb{R}_{\geq 0}^{Proc}$, for every delay transition $q_{\mathcal{M}_1} \xrightarrow{\boldsymbol{d}}_{\mathcal{M}_1} q'_{\mathcal{M}_1}$, there exists a matching delay transition $q_{\mathcal{M}_2} \xrightarrow{\boldsymbol{d}}_{\mathcal{M}_2} q'_{\mathcal{M}_2}$ such that $q'_{\mathcal{M}_1} \mathcal{R} q'_{\mathcal{M}_2}$ and symmetrically.

Two states $q_{\mathcal{M}_1}$ and $q_{\mathcal{M}_2}$ are multi-timed bisimilar, written $q_{\mathcal{M}_1} \approx q_{\mathcal{M}_2}$, iff there is a multi-timed bisimulation that relates them. $\mathcal{M}_1$ and $\mathcal{M}_2$ are multi-timed bisimilar, written $\mathcal{M}_1 \approx \mathcal{M}_2$, if there exists a multi-timed bisimulation relation $\mathcal{R}$ over $\mathcal{M}_1$ and $\mathcal{M}_2$ containing the pair of initial states.

As a consequence of Definition 3, the notion of multi-timed bisimulation extends to icTA and we have the following definition:

Definition 4. *Let $\mathcal{A}$ and $\mathcal{B}$ be two icTA. We say the automata $\mathcal{A}$ and $\mathcal{B}$ are multi-timed bisimilar, denoted $\mathcal{A} \approx \mathcal{B}$, iff $\forall\, \tau \in Rates$ MLTS($\mathcal{A}, \tau$) $\approx$ MLTS($\mathcal{B}, \tau$).*

When there is only one process, the multi-timed bisimulation is the usual timed bisimulation. Consider the two icTA $\mathcal{A}_p$ (top) and $\mathcal{A}_q$ (bottom) in Figure 2 (b) with the alphabet $\Sigma = \{a\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$ and $\tau = (t^2, 3t)$ i.e. $\tau_p(t) = t^2$ and $\tau_q(t) = 3t$. $\mathcal{A}_p$ and $\mathcal{A}_q$ in Figure 2 (b) depicts an icTA. $\mathcal{A}_p$ performs nondeterministically the transition with the guard $x^p \leq 2$, the action a, resets clock x^p to 0 and enters location s_1. Similarly, $\mathcal{A}_q$ performs nondeterministically the transitions with the guard $y^q \leq 2$, the action a, resets clock y^q to 0 and enters location t_1. We will show that these icTA are not multi-timed bisimilar (Definition 3) ever if their underling TA are bisimilar (and ever isomorphic): We have $(S_0, [x^p = 0], 0)$ in MLTS$(\mathcal{A}_p, \tau_p)$ and $(T_0, [y^q = 0], 0)$ since $\mathcal{A}_p$ can run the delay transition $(S_0, [x^p = 0], 0) \xrightarrow{(1,3)} (S_0, [x^p = 1.0], 1)$ and $\mathcal{A}_q$ in MLTS$(\mathcal{A}_q, \tau_q)$. We have $(S_0, [x^p = 0], 0) \not\approx (T_0, [y^q = 0], 0)$ can only match this transition with $(T_0, [y^q = 0], 0) \xrightarrow{(1,3)} (T_0, [y^q = 3], 1)$. From these states MLTS$(\mathcal{A}_p, \tau_p)$ can fire a while MLTS$(\mathcal{A}_q, \tau_q)$ cannot.

4.2 Decidability

Inspired by [12], we show that for given icTA $\mathcal{A}$, $\mathcal{B}$, checking whether $\mathcal{A} \approx \mathcal{B}$ is decidable via a suitable zone graph [12]. In order to define the notion of clock zone over a set of clocks X, we need to consider the set $\Phi^+(X)$ of extended clock constraints.

Definition 5. *A clock constraint ϕ is a conjunction of comparisons of a clock with a constant c, given by the following grammar, where ϕ ranges over $\Phi^+(X)$, $x_i, x_j \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, >, \leq, \geq, =\}$:*

$$\phi ::= true \mid x_i \sim c \mid x_i - x_j \sim c \mid \phi_1 \ \wedge \ \phi_2.$$

A clock constraint of the form $x_i - x_j \sim c$ is called diagonal constraint and x_i, x_j must belong to the same process. The notion of satisfaction of a clock constraint $\phi \in \Phi^+(X)$ by a valuation is given by the clause $\nu \models x_i - x_j \sim c$ iff $\nu(x_i) - \nu(x_j) \sim c$.

Informally, a clock zone $\mathcal{Z}$ is a conjunction of extended clock constraints $\phi \in \Phi^+(X)$ with inequalities of clock differences and its semantics is the set of clock valuations that satisfy it $[\![\mathcal{Z}]\!] = \{\nu \mid \nu \models \phi\}$. We omit the semantics brackets ($[\![\mathcal{Z}]\!]$) when obvious. For any clock zones $\mathcal{Z}$, $\mathcal{Z}'$ and finite set of clocks X, the semantics of the intersection, clock reset, inverse clock reset, time successor and time predecessor events on clock zone can be defined as: (i) $\mathcal{Z} \cap \mathcal{Z}' = \{\nu \mid \nu \in \mathcal{Z} \wedge \nu \in \mathcal{Z}'\}$, (ii) $\mathcal{Z} \downarrow_X = \{\nu[X \to 0] \mid \nu \in \mathcal{Z}\}$, (iii) $\mathcal{Z} \uparrow_X = \{\nu \mid \nu[X \to 0] \in \mathcal{Z}\}$, (iv) $\mathcal{Z} \uparrow = \{\nu +_\pi \boldsymbol{d} \mid \nu \in \mathcal{Z}$ *and* $\boldsymbol{d} \in \mathbb{R}_{>0}^{Proc}\}$, (v) $\mathcal{Z} \downarrow = \{\nu -_\pi \boldsymbol{d} \mid \nu \in \mathcal{Z}$ *and* $\boldsymbol{d} \in \mathbb{R}_{>0}^{Proc}\}$.

A zone graph [12] is similar to a region graph [2] with the difference that each node consists of pair (called a zone) of a location s and a clock zone $\mathcal{Z}$ (*i.e*, $q = (s, \mathcal{Z})$). For $q = (s, \mathcal{Z})$, we write $(s', \nu) \in q$ if $s = s'$ and $\nu \in \mathcal{Z}$, indicating that a state is included in a zone. Analogously, we can write $(s, \mathcal{Z}) \subseteq (s', \mathcal{Z}')$ to indicate that $s = s'$ and $\mathcal{Z} \subseteq \mathcal{Z}'$. We will use the notation $\mathsf{Action}(e)$ to denote the action a of the edge e. Furthermore, we extend the zone operations for an icTA $\mathcal{A}$ in the following way:

Definition 6. *Let $q = (s, \mathcal{Z})$ be a zone and $e = (s, a, \phi, Y, s') \in \rightarrow_{icta}$ be a transition of $\mathcal{A}$, then $\mathsf{post}(\mathcal{Z}, e) = \{\nu' \mid \exists \nu \in \mathcal{Z}, \exists \tau \in Rates, \exists t \in \mathbb{R}_{\geq 0}, (s, \nu, t) \xrightarrow{e}_{mlts(\mathcal{A},\tau)} (s', \nu', t)\}$ is the set of valuations that q can reach by taking the transition e.*

Definition 7. *Let $q = (s, \mathcal{Z}')$ be a zone and $e = (s, a, \phi, Y, s') \in \rightarrow_{icta}$ be a transition of $\mathcal{A}$, then $\mathsf{pred}(\mathcal{Z}', e) = \{\nu \mid \exists \nu' \in \mathcal{Z}', \exists \tau \in Rates, \exists t \in \mathbb{R}_{\geq 0}, (s, \nu, t) \xrightarrow{e}_{mlts(\mathcal{A},\tau)} (s', \nu', t)\}$ is the set of valuations that q can reach by executing the transition e.*

Intuitively, the zone $(s', \mathsf{post}(\mathcal{Z}, e))$ describes the discrete successor of the zone $(s, \mathcal{Z})$ under the transition e, and the zone $(s, \mathsf{pred}(\mathcal{Z}', e))$ describes the discrete predecessor of the zone $(s', \mathcal{Z}')$ under the transition e.

Definition 8 (Multi-timed Zone Graph). *Given an icTA $\mathcal{A} = (\Sigma, X, S, s_0, \rightarrow_{icta}, I, F, \pi)$, its symbolic multi-timed zone graph ($\mathsf{ZG}(\mathcal{A})$) is a transition system*

ZG($\mathcal{A}$) $= (Q, q_0, (\Sigma \cup \{\uparrow\}), \rightarrow_{ZG})$, *where :* (i) Q *consists of pairs* $q = (s, \mathcal{Z})$ *where* $s \in S$, *and* $\mathcal{Z} \in \Phi^+(X)$ *is a clock zone with* $\mathcal{Z} \subseteq I(s)$. (ii) $q_0 \in Q$ *is the initial zone* $q_0 = (s_0, \mathcal{Z}_0)$ *with* $\mathcal{Z}_0 = [\![\bigwedge_{x \in X} x = 0]\!]$. (iii) Σ *is the set of labels of* $\mathcal{A}$. (iv) $\rightarrow_{ZG} \subseteq Q \times (\rightarrow_{icta} \cup \{\uparrow\}) \times Q$ *is a set of transitions, where each transition in ZG($\mathcal{A}$) is a labelled by a transition* $e = (s, a, \phi, Y, s') \in \rightarrow_{icta}$, *where* s *and* s' *are the source and target locations,* ϕ *is a clock constraint defining the guard of the transition,* a *is the action of the edge and* Y *is the set of clocks to be reset by the transition in the icTA* $\mathcal{A}$. *For each* $e \in \Sigma$, *transitions are defined by the rules:*

(i) For every $e = (s, a, \phi, Y, s')$ *and clock zone* $\mathcal{Z}$, *there exists a discrete transition* (q, e, q'), *where* $q = (s, \mathcal{Z}) \xrightarrow{e}_{ZG} q' = (s', post(\mathcal{Z}, e))$ *if* $post(\mathcal{Z}, e) \neq \emptyset$.

(ii) For a clock zone $\mathcal{Z}$, *there exists a delay transition* $(q, \uparrow, q')$, *where* $q = (s, \mathcal{Z}) \xrightarrow{\uparrow}_{ZG} q' = (s, \mathcal{Z}')$ *and* $\mathcal{Z}' = \mathcal{Z}\uparrow \cap I(s)$.

Note that $\uparrow$ *is used here as a symbol to represent symbolic positive delay transitions. Only the reachable part is constructed.*

Lemma 1. *Let* $(s, \mathcal{Z})$ *be a zone and* $e = (s, a, \phi, Y, s') \in \rightarrow_{icta}$ *be a transition of an icTA* $\mathcal{A}$, *then* $\mathcal{Z}\uparrow$, $\mathcal{Z}\uparrow_x$, $\mathcal{Z}\downarrow$, $post(\mathcal{Z}, e)$ *and* $pred(\mathcal{Z}', e)$ *are also zones.*

Multi-timed Zone Graph Algorithm: In algorithm 1, we build a reachable multi-timed zone graph (ZG($\mathcal{A} \parallel \mathcal{B}$)) for the parallel composition of two icTA ($\mathcal{A}$ and $\mathcal{B}$). Algorithm 1 build a multi-timed zone graph, starting with the pair $(s_0, \mathcal{Z}_0)$ (s_0 initial location of the automaton $\mathcal{A}$ with $\mathcal{Z}_0 = [\![\bigwedge_{x \in X} x = 0]\!]$ represents the initial zone). However, the multi-timed zone graph can be infinite, because constants used in zones may grow for ever. Therefore, we use a technique called extrapolation abstraction ($Extra^+_{LU_{(s)}}$ (LU-bound)) [7][4], where L is the maximal lower bound and U is the maximal upper bounds. For every location s of a ZG($\mathcal{A}$), there are bound functions LU and the symbolic zone graph using $Extra^+_{LU_{(s)}}$. Then, we build zones of the form $q_{ZG} = (s, Extra^+_{LU_{(s)}}(post(\mathcal{Z}, e))$.

Lemma 2 (Completeness). *Let* $\theta = (s_0, \nu_0, t_0) \xrightarrow{\boldsymbol{d}_0, a_0} (s_1, \nu_1, t_1) \xrightarrow{\boldsymbol{d}_1, a_1} \ldots \xrightarrow{\boldsymbol{d}_{n-1}, a_{n-1}} (s_n, \nu_n, t_n)$ *be a run of MLTS($\mathcal{A}, \tau$), for some* $\tau \in Rates$. *Then, for any state* (s_i, ν_i, t_i) *where* $0 \leq i \leq n$, *there exists a symbolic zone* $(s_i, \mathcal{Z}_i)$ *added in* Q *such that* $\nu_i \in \mathcal{Z}_i$.

The above lemma tells that the algorithm 1 over-approximates reachability. Now, we can establish the termination of the algorithm 1, because there are finitely many $Extra^+_{LU_{(s)}}$ zones. Here, we will use algorithm 1 to over-approximate the co-reachable state space of the two icTA $\mathcal{A}$ and $\mathcal{B}$, on the strongly synchronized product of $\mathcal{A}$ and $\mathcal{B}$. The time complexity of this algorithm is given in terms of the number of clocks, the number of clocks and the number of transitions of the icTA: $O(|S| \times | \rightarrow_{\text{icTA}} | \times |X|^2))$ where $|S|$ represent the number of states in the

icTA $\mathcal{A}$, $|X|$ the number of clocks in $\mathcal{A}$ and $|\rightarrow_{\mathsf{icTA}}|$ the number of transitions in $\mathcal{A}$.

Algorithm 1: Reachable Multi-timed Zone Graph with subsumption

Input : An icTA $\mathcal{C} = (\Sigma, X, S, s_0, \rightarrow_{icta}, I, F, \pi)$.
Output: A reachable zone graph $\mathsf{ZG}(\mathcal{C}) = (Q, q_0, \Sigma, \rightarrow_{\mathsf{ZG}})$.

```
// s ∈ S is a location of C, Z_{1≤i≤3} are clock zones.
// T_ZG is a set of transitions (i.e. →_ZG= T_ZG), E_ZG is a set of labels.
// D and Q are a set of pairs S × Z, D is the set of open states.
Function BuildSymbZoneGraph(C)
    q_0 = (s_0, Z_0) such that for all x ∈ X and ν ∈ Z_0, ν(x) = 0 ;
    Q, D ← {q_0}, T_ZG ← ∅, M ← ∅ ;
    while D ≠ ∅ do
        Choose and Remove (s, Z_1) from D ;
        for each transition e = (s, a, φ, Y, s') ∈→_icta such that Z_1 ∧ φ ≠ ∅ do
            // Z_2 is the successor
            Z_2 ← Extra^+_{LU_(s)}(post(Z_1, e)) ;
            E_ZG ← E_ZG ∪ {e} ;
            if exists (s', Z_3) ∈ Q such that Z_2 ⊆ Z_4 then
                T_ZG ← T_ZG ∪ {(s, Z_1) -e->_ZG (s', Z_3)} ;
            else
                T_ZG ← T_ZG ∪ {(s, Z_1) -e->_ZG (s', Z_2)} ;
                Q ← Q ∪ {(s', Z_2)}, D ← D ∪ {(s', Z_2)} ;
            end
        end
        Z_2 ← Z_1 ↑ ∧ I(s) ;
        if exists (s, Z_3) ∈ Q such that Z_2 ⊆ Z_3 then
            T_ZG ← T_ZG ∪ {(s, Z_1) -↑->_ZG (s, Z_3)} ;
        else
            T_ZG ← T_ZG ∪ {(s, Z_1) -↑->_ZG (s', Z_2)} ;
            Q ← Q ∪ {(s, Z_2)}, D ← D ∪ {(s, Z_2)} ;
        end
    end
    return (Q, q_0, Σ, →_ZG) ;
end
```

Refinement Algorithm: Now, we describe a refinement algorithm with signature to compute the multi-timed bisimulation from their zone graph of their strong product $\mathsf{ZG}(\mathsf{ZG}(\mathcal{A} \parallel \mathcal{B}))$. The passage of arbitrary local times are abstracted by time elapse $\uparrow$ transitions from a zone to successor zones, and discrete transitions. Essentially, our algorithm is based on the refinement technique [17][19][6]. The state space Q of $\mathsf{ZG}(\mathcal{A} \parallel \mathcal{B})$ is divided in zones that initially over-approximate the co-reachable states of $\mathcal{A}$ and $\mathcal{B}$). The algorithm 2 start from an initial set of zones Π_0 and successively refines these sets such that ultimately each zone contains only bisimilar state pairs.

The runs of a zone graph involve a sequence of moves with discrete and time-elapse $\uparrow$ transitions. The refinement algorithm has thus to deal with the following difficulties: when taking a $\uparrow$ transition, where the clocks in different processes are not perfectly synchronous, it should take into consideration that the time elapse traverses continuously diagonal, almost vertical and horizontal time successor zones. Conversely, when the clocks belonging to the same process (i.e., perfectly synchronous), the time elapsing traverses only continuously diagonal time successor zones. Thus, the time refinement operator presented in [19] is not applicable within our algorithm 2. Figure 3 presents an example : (a) a time

elapsing traversing the clock regions 1 to 3 for synchronous clocks, (b) a time elapsing traversing continuously diagonal, almost horizontal and vertical time successor zones for asynchronous clocks.

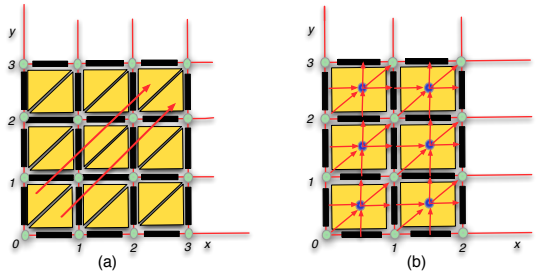

Fig. 3. (a) A time elapsing traversing 0 to 3, (b) Multi-timed time successors.

The discrete refinement operator presented in [19] is also not applicable within our algorithm 2. Therefore, our algorithm adopts the idea of the signature-based technique [6], which assigns states to equivalence blocks according to a characterizing signature. In each refinement iteration, the set of zones are refined according to a signature. The algorithm in [6], cannot be applied in our setting in a straightforward way, due to its untimed characteristic, while in our case, the time and discrete characteristics should be considered. Based on [6], we introduce a signature refinement operator which refine the set of zones until a fixed point is reached, which is the complete multi-timed bisimulation. Thus, we introduce the timed and discrete predecessor operators.

Definition 9. *Let* $q = (s, \mathcal{Z})$ *and* $q' = (s, \mathcal{Z}')$ *be two zones, then :* $\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}') = \{\nu \in \mathcal{Z} \mid \exists\ \boldsymbol{d} \in \mathbb{R}^{Proc}_{>0},\ \exists\ \tau \in Rates,\ \exists\ t, t'' \geq 0,\ \ t \leq t''$ *and* $\forall t', t \leq t' \leq t''$, *and* $\boldsymbol{d} = \tau(t'') - \tau(t),\ (\nu +_{\pi} \boldsymbol{d}) \in\ \mathcal{Z}'$, *and* $\boldsymbol{d'} = \tau(t') - \tau(t)$ *then* $(\nu +_{\pi} \boldsymbol{d'}) \in (\mathcal{Z} \cup \mathcal{Z}')\}$ *is the set of valuations in the zone* $\mathcal{Z}$ *from which a valuation of* $\mathcal{Z}'$ *can be reached through the elapsing of time, without entering any other zones besides* $\mathcal{Z}$ *and* $\mathcal{Z}'$ *(i.e.,* $\mathcal{Z} \cup \mathcal{Z}'$*).*

The $\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$ operator refines $\mathcal{Z}$ selecting the states that can reach $\mathcal{Z}'$.

Lemma 3. *Let* $q = (s, \mathcal{Z})$, $q' = (s, \mathcal{Z}') \in Q$ *be two zones, then* $\mathsf{TimePred}_{\uparrow}(\mathcal{Z}, \mathcal{Z}')$ *is a clock zone.*

We use as signature of a state (s, ν) the set of outgoing transitions from (s', ν'). Then, a refinement of a zone can be computed by grouping states that have the same signature. The resulting set of zones then represents the multi-timed bisimulation relation: two states (s, ν) and (s', ν') are multi-timed bisimilar iff they are in the same zone with similar outgoing transitions. Formally, this is captured in the following definition:

Definition 10. *Let* $q = (s, \mathcal{Z})$ *be a zone, then the signature of a state* $(s, \nu) \in q$ *formed by the set of labels of all the edges starting from* (s, ν) *is defined as:*

$\mathsf{ActionSigPred}_q(s,\nu) = \{(Action(e)) \mid \exists \mathcal{Z}',\ \exists\ \nu' \in \mathcal{Z}',\ (s,\nu) \xrightarrow{Action(e)}_{icTA} (s',\nu')\}$. *Also, the signature of the zone* q *is defined as:* $\mathsf{ActionSig}(q) = \bigcup_{(s,\nu)\in q} \mathsf{ActionSigPred}_q(s,\nu)$.

$\mathsf{ActionSigPred}_q(s,\nu)$ operator is used to compute the signatures of a state into a zone. Our algorithm 2 consists of two steps: The initial phase, is responsible for keeping a pair of states in q into zones so that every pair of states ($i.e, ((s_\mathcal{A}, s_\mathcal{B})$ $, (\nu_\mathcal{A}, \nu_\mathcal{B}))$) from the same zone q have the same signature $\mathsf{ActionSigPred}_q(s_\mathcal{A}, \nu_\mathcal{A})$ $= \mathsf{ActionSigPred}_q(s_\mathcal{B}, \nu_\mathcal{B})$. The refinement phase, consists of computing the timed predecessors (see Definition 11 below) and the discrete signature predecessors (see Definition 12 below) until a stable set of zones is reached. Stable zone are a multi-timed bisimulation relation if every pair of states of every zone in the set have the same signature with respect to every computed refinement. A detailed explication about building a stable zones follows:

- Initial phase: Let $\Pi_0 = Q$ be the initial set of zones, where Q is given by algorithm 1. After the initial phase, the set Π contains zones consisting of states with unique signatures, $\mathsf{ActionSigPred}_q(s_\mathcal{A}, \nu_\mathcal{A}) = \mathsf{ActionSigPred}_q(s_\mathcal{B}, \nu_\mathcal{B})$.

- Refinement phase: An existing set of zones are iteratively refined until all zones becomes stable simultaneously with respect to all their timed predecessors and discrete predecessors. For simplicity, we will write $(s, \mathcal{Z})$ to denote the pairs $((s_\mathcal{A}, s_\mathcal{B}), \mathcal{Z})$.

Definition 11. *Let* Π *be a set of zones and* $q = (s, \mathcal{Z})$, $q' = (s', \mathcal{Z}')$ *be two zones in* Π. *Then for the delay transitions, the refinement function is defined as follows:*

$\mathsf{TimeRefine}(\mathcal{Z}, \Pi) = \{\mathsf{TimePred}_\uparrow(\mathcal{Z}, \mathcal{Z}') \mid \mathcal{Z}' \in \Pi, q \xrightarrow{\uparrow}_\Pi q'\}$.

Definition 12. *Let* Π *be a set of zones and* $q = (s, \mathcal{Z})$, $q' = (s', \mathcal{Z}')$ *be two zones in* Π. *Let* $q = (s, \mathcal{Z})$ *be the currently examined zone and* $\mathsf{ActionSig}(q)$ *be the signatures of the set of states into the zone* q. *Let* $e_\mathcal{A}$ *and* $e_\mathcal{B}$ *be the transitions of the* icTAs $\mathcal{A}$ *and* $\mathcal{B}$. *Then the refinement of a zone* q *is defined as follows:*

$\mathsf{DiscreteSigRefine}(\mathcal{Z}, \Pi) = \bigcap_{a\in \mathsf{ActionSig}(q)} ((\bigcap_{\{e_\mathcal{A} \mid Action(e_\mathcal{A})=a\}} \bigcup_{\{e_\mathcal{B} \mid Action(e_\mathcal{B})=a\}} \mathsf{pred}(\mathcal{Z}', (e_\mathcal{A}, e_\mathcal{B}))) \cap (\bigcap_{\{e_\mathcal{B}\ \mid\ Action(e_\mathcal{B})=a\}} \bigcup_{\{e_\mathcal{A}\ \mid\ Action(e_\mathcal{A})=a\}} \mathsf{pred}(\mathcal{Z}', (e_\mathcal{A}, e_\mathcal{B}))))$.

Lemma 4. *Let* $(s, \mathcal{Z})$ *be a class of* Π *and let* e *be an edge of the* $\mathsf{ZG}(\mathcal{C})$, *then each of* $\mathsf{TimeRefine}(\mathcal{Z}, \Pi)$ *and* $\mathsf{DiscreteSigRefine}(\mathcal{Z}, \Pi)$ *forms a partition of* $\mathcal{Z}$ *in zones.*

The correctness of the algorithm 2 follows from the algorithm in [17] and [6]. The definition $\mathsf{TimeRefine}(\mathcal{Z}, \Pi)$ above to generate a finer set of zones, which deals with delay transitions. The definition of $\mathsf{DiscreteSigRefine}(\mathcal{Z}, \Pi)$, generate also a finer set of zones and distinguishes the states with discrete transitions.

Termination is ensured by Lemma 4. Algorithm 2 describes the main steps of the decision procedure for multi-timed bisimulation checking. It is based on the function BuildSymbZoneGraph (i.e., algorithm 1). The function PartitionZoneGraph returns stable set of zones Π. Given a set of zones Π, the algorithm 2 computes the states $((s_{\mathcal{A}}, s_{\mathcal{B}}), \mathcal{Z})$ from Π that are bisimilar up to the desired initial state $((s^0_{\mathcal{A}}, s^0_{\mathcal{B}}), \mathcal{Z}_0)$.

Algorithm 2: The partition refinement algorithm for a reachable ZG

Input : A $\mathsf{ZG}(\mathcal{C}) = (Q = Q_{\mathcal{A}} \times Q_{\mathcal{B}}, q_0 = (q^0_{\mathcal{A}}, q^0_{\mathcal{B}}), \Sigma = \Sigma_{\mathcal{A}} \cup \Sigma_{\mathcal{B}}, \rightarrow_{\mathsf{ZG}})$, Π.
Output: A coarsest partition Π.

```
// q ∈ Q is a zone of ZG(C), Π is a set of zones, Z, Z' are clock zones.
// Q is a set of pairs S × Z.
Function PartitionZoneGraph(ZG(C), Π)
    // Phase I - Get the input partition Π
    Π' ← Π ;
    Repeat
        // Phase II - Refine Π' by delay transitions:
        for each zone (or block) Z ∈ Π' do
            Π' ← TimeRefine(Z, Π') ;
        end
        // Phase III - Refine Π' by discrete transitions:
        for each zone (or block) Z ∈ Π' do
            Π' ← DiscreteSigSplit(Z, Π') ;
        end
    Until Π' does not change;
    Return Π' ;
end
```

Proposition 1. *Let $q = (s, \mathcal{Z})$ be a zone. Let $(s_{\mathcal{A}}, \nu_{\mathcal{A}})$ and $(s_{\mathcal{B}}, \nu_{\mathcal{B}})$ be two states in q, then $(s_{\mathcal{A}}, \nu_{\mathcal{A}}) \approx (s_{\mathcal{B}}, \nu_{\mathcal{B}})$ iff $((s_{\mathcal{A}}, s_{\mathcal{B}}), \nu_{\mathcal{A}} \cup \nu_{\mathcal{B}}) \in \mathcal{Z}$.*

Theorem 1. *Deciding multi-timed bisimulation between two icTA is EXPTIME-complete.*

An example of the zone graph, partition and multi-timed bisimulation computed by our algorithms can be found in Figure 4. The Figure 4 (a) shows two icTA $\mathcal{A}$ and $\mathcal{B}$ with the finite input alphabet $\Sigma = \{a, b\}$, the set of processes $Proc = \{p, q\}$, the set of clocks $X = \{x^p, y^q\}$ and $\tau_p > \tau_q$. The Figure 4 (b) shows the zone graph computed by algorithm 1. The Figure 4 (c) shows the multi-timed bisimualtion for $\mathcal{A}$ and $\mathcal{B}$.

5 Related Work

Because TA are a general-purpose formalism, several implementations and extensions have been considered. For example, Puri [18] studied the semantics of robustness timed automata where clocks can drift in a bounded way, i.e. clocks may grow at independent rates in the interval $1 \pm \epsilon$. Krishnan [11] considered asynchronous distributed timed automata, where clocks evolve independently in each component. Akshay *et al.* concentrate on the untimed language of DTA. In a previous work [16], we suggested a model that has the same expressive power as event clock automata [2], but without studied possible simulation algorithms.

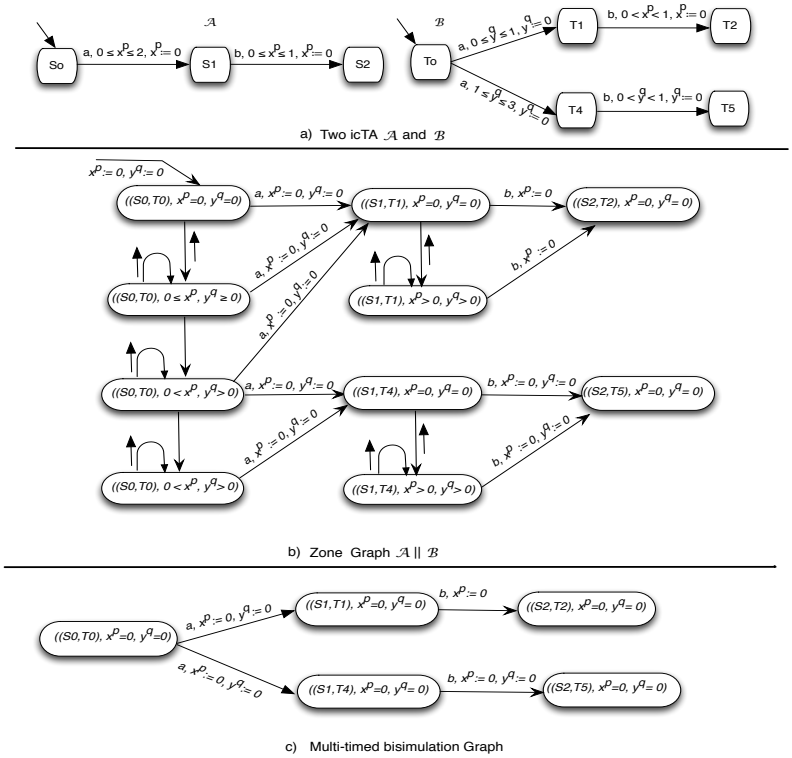

Fig. 4. (a) Composition of icTAs; (b) Zone graph; (c) bisimulation

The notion of bisimulation for TA is studied in various contributions [9, 20, 19, 8, 4]. Cerans [9] gives a proof of decidability for timed bisimulation. Several techniques are used in the literature for providing algorithms capable of checking (bi-)simulation: Weise and Lenzkes [20] rely on a zone-based algorithm for weak bisimulation over TA, but no implementation is provided; Bulychev *et al.* [8] study timed simulation for simulation-checking games, for which an implementation is available from [4]; region construction for timed bisimulation was also considered by Akshay *et al.* [1], but never implemented; and more closely to our work, Tripakis and Yovine proposed a time-abstract bisimulation over TA in [19]. Krishnan [11] and our previous work [16] manipulated clock drifts as well for manipulating DTA, but without considering bisimulation.

6 Conclusions

Bisimulation is a common technique to reduce the state space explosion issue encountered during model-checking of real-time systems. To enable the application of this technique for DTS modelled by icTA, we proposed an alternative

semantics for capturing the execution of icTA, based on multi-timed words running over Multi-Timed Labelled Transition Systems. We extended the notion of bisimulation to such structures, and proposed an EXPTIME algorithm for checking decidability. We are now studying how to efficiently implement such structures and decidability algorithm, and plan to compare their performance against classical work as proposed in [4, 19].

References

1. S. Akshay, B. Bollig, P. Gastin, M. Mukund, and K. N. Kumar. Distributed timed automata with independently evolving clocks. In *CONCUR*, LNCS. Springer, 2008.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 1994.
3. S. Balaguer and T. Chatain. Avoiding shared clocks in networks of timed automata. *Logical Methods in Computer Science*, 2013.
4. G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *STTT*, 2006.
5. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lecture Notes on Concurrency and Petri Nets*, 2004.
6. S. Blom and S. Orzan. A distributed algorithm for strong bisimulation reduction of state spaces. *Electr. Notes Theor. Comput. Sci.*, 2002.
7. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.
8. P. Bulychev, T. Chatain, A. David, and K. G. Larsen. Efficient on-the-fly algorithm for checking alternating timed simulation. In *FORMATS*, LNCS. Springer, 2009.
9. K. Cerans. Decidability of bisimulation equivalences for parallel timer processes. In *CAV*, London, UK, 1993.
10. M. De Biasi, C. Snickars, K. Landernäs, and A. Isaksson. Simulation of process control with wirelesshart networks subject to clock drift. In *COMPSAC*, 2008.
11. P. Krishnan. Distributed timed automata. *In Workshop on Distr. Systems*, 1999.
12. F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic - and back. In *MFCS*, LNCS. Springer, 1995.
13. R. Milner. *Communication and concurrency*. Prentice Hall, 1989.
14. A. Monot, N. Navet, and B. Bavoux. Impact of clock drifts on CAN frame response time distributions. In *ETFA*, Toulouse, France, 2011.
15. J. Ortiz and P.-Y. Schobbens. Extending Timed Bisimulation for Distributed Timed Systems. Technical report, University of Namur, 2016. http://www.info.fundp.ac.be/∼jor/Multi-TimedReport/.
16. J. J. Ortiz, A. Legay, and P.-Y. Schobbens. Distributed event clock automata. In *CIAA*, LNCS. Springer, 2011.
17. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, pages 973–989, 1987.
18. A. Puri. Dynamical properties of timed automata. In A. P. Ravn and H. Rischel, editors, *FTRTFT*, volume 1486 of *LNCS*, pages 210–227. Springer, 1998.
19. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 2001.
20. C. Weise and D. Lenzkes. Efficient scaling-invariant checking of timed bisimulation. In *STACS*, LNCS. Springer, 1997.